1 Predictive Location Tracking in Cellular and in Ad Hoc Wireless Networks

NIKOS DIMOKAS[1], DIMITRIOS KATSAROS[1,2],
PANAYIOTIS BOZANIS[2] and YANNIS MANOLOPOULOS[1]

[1]Department of Informatics, Aristotle University of Thessaloniki, Greece
[2]Department of Computer & Communication Engineering, University of Thessaly, Greece

Predicting the future is mostly a matter of managing not to blink as you witness the present.
—William Gibson

1.1 INTRODUCTION

The proliferation of cellular and ad hoc networks and the penetration of Internet services are changing many aspects of ubiquitous mobile computing. Constantly increasing mobile client populations utilize diverse mobile devices to access the wireless medium and various heterogeneous applications (e.g., streaming video, Web) are being developed to satisfy the eager client requirements. The realization of such a demanding environment requires addressing many technical issues, related to radio management, networking, data management and so on.

Most of the challenging issues and problems in this area are due to the fact that the underlying environment is extremely resource-starving and inherently uncertain. For instance, the wireless communication channels are bandwidth-limited and error prone. The uncertainty due to node (user) mobility has fundamental impacts, since it induces uncertainly in the network topology and hence causes problems in routing and in data delivery. Additionally, traffic load and resource demands in cellular and in ad hoc wireless networks are also uncertain, depending a lot on the user trajectories.

In this harsh environment, seamless and ubiquitous connectivity is a fundamental goal. This goal calls for smart techniques for determining the current and future location of a mobile. The ability to determine the mobile client's (future) location can significantly improve the wireless network's overall performance. Consider for

instance the handover procedure in cellular networks, which is directly related to the design of resource management algorithms in such infrastructured networks; such resources could be bandwidth, MAC frames, packets. Instead of relying on *reactive* approaches, i.e., allocating appropriate resources during the handover, we could come up with *proactive* approaches, i.e., allocating resources before needed, so as to bypass, instead of correct, the negative effect of handover [28]. Additionally, methods like the Shadow Cluster [31], could benefit from location prediction, by refraining from allocating resources to all neighboring cells; with the exploitation of predictions instead, they could allocate resources only to the most probable-to-move cells. Finally, location prediction could be exploited in sequential paging schemes [8] to reduce the combined paging cost and also in techniques for call admission control [60].

Location prediction and tracking is useful not only in cellular networks, but to other types of wireless networks as well, such as mobile ad hoc networks (MANETs). A mobile ad hoc network is a wireless network in which a set of mobile nodes with wireless connectivity form a temporary network without the existence and support of any infrastructure, e.g., base stations, or centralized administration, e.g., switching centers. Communication in an ad hoc network between any two nodes that are out of one another's transmission range is achieved through intermediate nodes, which relay messages to set up a communication channel between the two nodes. For a MANET node v wishing to communicate with another MANET node u not within its transmission range, knowledge of the future position(s) of node v could help reduce its energy consumption, by postponing its communication to u until it reaches closer to it. Such a technique has been investigated in [9].

1.1.1 Preliminaries

The present chapter deals with the issue of predictive location tracking in cellular and in ad hoc wireless networks, and examines this issue in two different settings. In section 1.2, we assume a generic symbolic network topology model, like that introduced in [8], where the existence of "cells" is assumed. The cells need not be hexagonal, but can be of arbitrary shape. The notion of a wireless cell is well established in cellular networks; in ad hoc networks can be defined in a similar manner, if we overlay a grid of any type [9] over the area, inside which the mobile hosts of the wireless ad hoc network are roaming. In this setting, the positioning of a mobile is performed at the level of a cell. In section 1.3, we withdraw the assumption of the existence of cells, and the position of each mobile host is determined by its geographic coordinates only.

Connectivity in the presence of mobility is a nontrivial task; the network has to work against the uncertainty created by the mobile's freedom of movement. Thus, the management of mobility is of crucial importance. Depending on whether a mobile terminal is actively communicating or in standby mode, we differentiate between a) in-session mobility management, and b) out-of-session mobility management. The former is widely known in cellular networks as *handoff* management, and it deals with mechanisms by which calls and sessions are kept alive while the mobile host is

moving from cell to cell, thus changing its network point of attachment. In general, the procedure of handoff management is considered easier than the management of the latter case, which is known as *location management* or *location tracking* and which is responsible for keeping track of mobiles in standby mode.

The location tracking problem in generic wireless networks involves two procedures, namely *paging* and *update*. At the one extreme, one can come up with a proposal for this problem with the aid of paging, which is performed by the system; on a call arrival, the network initiates a search for the sought mobile, by (simultaneously) polling every possible site where the mobile can be found. In the case of cellular networks, this is performed by the *mobile switching center* which broadcasts a page message over a special *forward control channel* via the *base stations*. All the mobiles listen this paging message, but only the target mobile responses over a *reverse channel*. In the worst case, the system may have to page each cell of the whole service area. Clearly, this approach involves excessive signaling traffic and thus is problematic.

At the other extreme, one can come up with a solution, which demands from the mobile to report every time it moves from one site (cell) to another. This reporting is called *location registration*, and starts with an update message sent by the mobile over a reverse channel, which is then followed by some traffic that takes care of related database maintenance operations at the system's side. Again, this approach may also generate excessive signaling traffic if the mobile changes cells frequently and thus it is impractical.

In real situations location tracking is performed as a hybrid between these extreme approaches [47]. Although a lot of (reactive) location management methods have been proposed, the issue of predictive (or proactive) location, tracking has lately received significant attention, due to its potential to reduce or even eliminate the latency associated to location tracking. Moreover, there are situation where prediction of mobiles' movements, that will eventually lead to network disconnection, may force specific decisions related to routing. Examples of these decisions involve the routing protocols which are suitable for highly mobile ad hoc networks and for delay-tolerant networks [62].

In general, predictive location tracking techniques work by constructing a *mobility, model* for each mobile host that models the *mobility history* of the mobile. Clearly, the two notions are different; the former is probabilistic and extends to the future, whereas the latter is deterministic and refers to the past. Location prediction is related to the ability of the underlying network to record, learn and, subsequently *predict* the mobile's movements. The success of the prediction is presupposed and is boost by the fact that *mobile users exhibit some degree of regularity* in their movement [8]. A "smart" network can record the movement history and then construct a mobility model for its clients. The real challenge involved in designing an effective and efficient predictive location tracking method is to quantify the utility of the past in predicting the future.

1.1.2 Chapter Organization

The purpose of this chapter is to provide an overview of techniques suitable for predicting the future locations of mobile hosts in wireless networks. It concentrates on two different scenarios; according to the first scenario, the network coverage area is partitioned in non-overlapping regions (named cells) and the location tracking is performed at the level of cells; according to the second scenario, there is no such tiling to the coverage area and location tracking is performed at the level of geographical coordinates. The first part of the chapter (Section 1.2) deals with the first scenario and it introduces information-theoretic methods suitable for predicting the future locations of mobiles. The second part of the chapter (Section 1.3) deals with the second scenario and presents the issues related to indexing the positions of mobile hosts in order to support predictive queries. For these two broad and significant issues, the chapter surveys, classifies, and compares the state-of-the-art solutions, by discussing the critical issues and challenges of predictive location tracking in wireless networks.

1.2 PREDICTIVE LOCATION TRACKING TECHNIQUES

Being the uncertainty inherent in the mobile's movements we can consider them to be the outcome of an underlying stochastic process, which can be modelled using established information-theoretic concepts and tools [56, 34]. The cornerstone work of [17] exhibited the possibility of using methods, which have traditionally been used for data compression (thus, characterized as "information-theoretic"), in carrying out prediction. Considering a symbolic network topology model [8], we can model the respective state space as a finite alphabet comprised of discrete symbols. The alphabet consists of all possible sites (cells) where the client has ever visited or might visit (assuming that the number of cells in the coverage area is finite). With this transform, we can exploit methods, which have traditionally been used for data compression (thus, characterized as "information-theoretic"), to carry out prediction. In the rest of this section, we elaborate on these methods.

1.2.1 The discrete sequence prediction problem

In quantifying the utility of the past in predicting the future, a formal definition of the problem is needed, which we provide in the following lines. Let Σ be an alphabet, consisting of a finite number of symbols $s_1, s_2, \ldots, s_{|\Sigma|}$, where $|\cdot|$ stands for the length/cardinality of its argument. A *predictor*, which is an algorithm used to generate prediction models, accumulates sequences of the type $\mathfrak{a}_i = \alpha_i^1, \alpha_i^2, \ldots, \alpha_i^{n_i}$, where $\alpha_i^j \in \Sigma, \ \forall i, j$ and n_i denotes the number of symbols comprising $\mathfrak{a}_i$. Without loss of generality, we can assume that all the knowledge of the predictor consists of a single sequence $\mathfrak{a} = \alpha^1, \alpha^2, \ldots, \alpha^n$. Based on $\mathfrak{a}$, the predictor's goal is to construct a model that assigns probabilities for any future outcome given "some" past. Using

the characterization of the mobility model as a stochastic process $(\mathbb{X}_t)_{t\in N}$, we can formulate the aforementioned goal as follows:

Definition 1 (Discrete Sequence Prediction problem). *At any given time instance t (meaning that t symbols $x_t, x_{t-1}, \ldots, x_1$ have appeared, in reverse order) calculate the conditional probability*

$$\widetilde{P}[\mathbb{X}_{t+1} = x_{t+1} | \mathbb{X}_t = x_t, \mathbb{X}_{t-1} = x_{t-1}, \ldots],$$

where $x_i \in \Sigma,\ \forall x_{t+1} \in \Sigma$. This model introduces a stationary Markov chain, since the probabilities are not time-dependent. The outcome of the predictor is a ranking of the symbols according to their $\widetilde{P}$. The predictors which use such kind of prediction models are termed Markov predictors.

Depending on the application, the predictor may return only the symbol(s), with the highest probability, i.e., implementing a "most-probable" prediction policy, or the symbols with the m highest probabilities, i.e., implementing a "top-m" prediction policy, where m is an administratively set parameter. In any case, the selection of the policy is a minor issue and will not be considered in this paper, which is only concerned with methods for inferring the ranking.

The "history" $x_t, x_{t-1}, \ldots$ used in the above definition is called the *context* of the predictor, and it refers to the portion of the past that influences the next outcome. The history's length (also, called the *length* or *memory* or *order* of the Markov chain/predictor) will be denoted by l. Therefore, a predictor which exploits l past symbols, will calculate conditional probabilities of the form

$$\widetilde{P}[\mathbb{X}_{t+1} = x_{t+1} | \mathbb{X}_t = x_t, \mathbb{X}_{t-1} = x_{t-1}, \ldots, \mathbb{X}_{t-l+1} = x_{t-l+1}]. \qquad (1.1)$$

Some Markov predictors fix, in advance of the model creation, the value of l, presetting it in a constant k, in order to reduce the size and complexity of the prediction model. These predictors, and the respective Markov chains are termed *fixed length Markov chains/predictors* of order k. Therefore, they compute probabilities of the form:

$$\widetilde{P}[\mathbb{X}_{t+1} = x_{t+1} | \mathbb{X}_t = x_t, \mathbb{X}_{t-1} = x_{t-1}, \ldots, \mathbb{X}_{t-k+1} = x_{t-k+1}]. \qquad (1.2)$$

where k is a constant.
Although it is a nice model from a probabilistic point of view, these Markov chains are not very appropriate from the estimation point of view. Their main limitation is related to their structural poverty, since there is no means to set an optimized value for k.

Other Markov predictors deviate from the fixed memory assumption, and allow the order of the predictor to be of variable length, i.e., to be a function of the values from the past.

$$\widetilde{P}[\mathbb{X}_{t+1} = x_{t+1} | \mathbb{X}_t = x_t, \mathbb{X}_{t-1} = x_{t-1}, \ldots, \mathbb{X}_{t-l+1} = x_{t-l+1}], \qquad (1.3)$$

where $l = l(x_t, x_{t-1}, \dots)$.
These predictors are termed *variable length Markov chains*; the length l might range from 1 to t. If $l = l(x_t, x_{t-1}, \dots) \equiv k$ for all $x_t, x_{t-1}, \dots$, then we obtain the fixed length Markov chain. The variable length Markov predictors may or may not impose an upper bound on the considered length. The concept of variable memory offers a richness in the prediction model and the ability to adjust itself to the data distribution. If we can choose in a data-driven way the function $l = l(\cdot)$, then we can only gain with respect to the ordinary fixed length Markov chains, but this is not a straightforward problem.

The Markov predictors (fixed or variable length) base their probability calculations $\widetilde{P}$ on counts of the number of appearances of symbols after contexts. They also take special care to deal with the cases of unobserved symbols (i.e., symbols with zero appearance counts after contexts), assigning to them some "minimum probability mass".

1.2.2 The power of Markov predictors

The issue of prediction in wireless networks, especially location prediction, has received attention during the past years, and the most important proposed techniques focus around the notions of *learning automata*, *Kalman filtering* and *pattern matching*.

Learning automata are finite state adaptive systems that interact continuously with their environment learning a "behavior". Learning automata have been used in location prediction [28], and although they are simple, they are not considered very efficient learners, because of the need to devise appropriate penatly/reward policies, which is usually done in an ad hoc way, and due to their slow convergence to the correct actions.

Kalman filtering is a recursive processing algorithm for producing optimal estimates. Kalman filtering-based methods [33] construct a mobile motion equation relying on specific distributions for its velocity, acceleration and direction of movement. Therefore, they assume relatively accurate geographic position knowledge, via signal strength measurements. Their performance largely depends on the stabilization time of the Kalman filter and knowledge (or estimation) of the system's parameters.

Finally, (approximate) pattern matching techniques have been used for location prediction [33]. They compile (or assume existence of) aggregate or per-user mobility profiles, and perform approximate similarity matching between the current and the stored trajectories. The similarity matching is carried out through the computation of the *edit distance* between the current and each stored trajectory in order to derive predictions. Although edit distance computation can be performed quite fast with dynamic programming, it is relatively hard to select the meaningful set of edit operations on the individual symbols (i.e., insert, delete, substitute), to assign weights on them, deal with unequal sequences of symbols, select as similarity metric the edit distance instead of string alignment.

Consequently, a couple of questions arise regarding a) why Markov predictors are more appropriate for carrying out location prediction from the technical viewpoint, and b) whether location prediction are amenable to Markovian prediction. Several technical reasons advocate the use of Markov predictors in these problems, but their most profound advantage is their generality; they are domain independent without any coupling to geographic coordinates, or particular assumptions on distributions. A simple mapping from the "entities" of the investigated domain to an alphabet is all that is required. Thus, they are able to support location prediction.

Markovian prediction relies on the *short memory principle*, which, simply stated, says that the (empirical) probability distribution of the next symbol, given the preceding sequence, can be quite accurately approximated by observing no more than the last few symbols in that sequence. This principle fits reasonably and intuitively with how humans are acting when travelling or seeking information. A mobile user usually travels with a specific destination in mind, designing its travel via specific routes (roads or preferred pedestrian paths). This "targeted" traveling is far from a random walk assumption, and it is confirmed by studies with real mobility traces [45]. Therefore, the power of Markovian prediction stems from its generality and modelling capability, and also from its natural accordance with the human behavior.

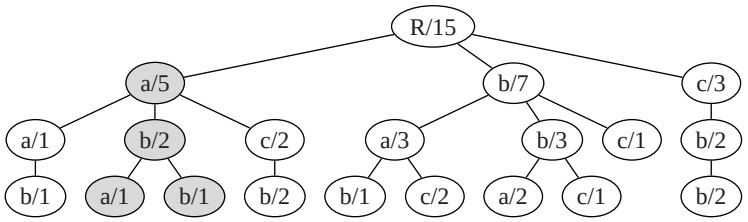
1.2.3 Families of Markov predictors

Markov predictors create probabilistic models for their input sequence(s) and they use *digital search trees (tries)* to keep track of the contexts of interest, along with some counts used in the calculation of the conditional probabilities $\widetilde{P}$. The root node of the trie corresponds to the "null" event/symbol, whereas every other node of the tree corresponds to a sequence of events; the sequence is used to label the node. We will consider a Markov predictor to be equivalent to its respective trie. Each node is accompanied by a counter, which depicts how many times this event has appeared after the sequence of events corresponding to the path from the root to the node's father has been observed.

For our convenience, we present some definitions useful in the sequel of the paper. We use the sample sequence of events $\mathfrak{a} = aabacbbabbacbbc$. The *length* of $\mathfrak{a}$ is the number of symbols it contains, i.e., $|\mathfrak{a}| = 15$. The *appearance count* of subsequence $\mathfrak{s} = ab$ is $E(\mathfrak{s}) = E(ab) = 2$ and the *normalized appearance count* of $\mathfrak{s}$ is equal to $E(\mathfrak{s})$ divided by the maximum number of (possibly overlapping) occurrences a subsequence of the same length could have, considering the $\mathfrak{a}$'s length, i.e., $E_n(\mathfrak{s}) = \frac{E(\mathfrak{s})}{|\mathfrak{a}|-|\mathfrak{s}|+1}$. The *conditional probability* of observing a symbol after a given subsequence, is defined as the number of times that symbol has shown up right after the given subsequence divided by the total number of times that the subsequence has shown up a all, followed by any symbol. Therefore, the conditional probability of observing the symbol b after the subsequence a will be denoted as $\widetilde{P}(b|a)$ and it is equal to $\widetilde{P}(b|a) = \frac{E(ab)}{E(a)} = 0.4$. In the rest of this section, we survey the families of Markov predictors.

1.2.3.1 The Prediction by Partial Match scheme The Prediction by Partial Match scheme, $\mathcal{PPM}$ for short, is based on the universal compression algorithm reported in [14]. For the construction of the prediction model, it assumes a pre-determined maximal order, say k, for the generated model. Then, for every possible subsequence of length of 1 up to $k+1$, creates or updates the appropriate nodes in the trie. Although, this description implies that the whole input sequence is known in advance, the method works in an online fashion, by exploiting a "sliding" window of size $k+1$ over the sequence as this grows symbol by symbol. The $\mathcal{PPM}$ predictor for the sample sequence $aabacbbabbacbbc$ is depicted in Figure 1.1. We can compute the conditional probability of a symbol σ to appear after a context s, by detecting the sequence $s\sigma$ as a path in the trie emanating from the root, provided that $|s\sigma| \leq k$. Prediction is performed in a similar manner. For instance, adopting a "most probable" prediction policy, the predicted symbol for the test context ab is a or b and its conditional probability is 0.50 for either of them. (See the gray-shaded nodes in Figure 1.1.)

Fig. 1.1 A $\mathcal{PPM}$ Markov predictor for the sequence $aabacbbabbacbbc$.



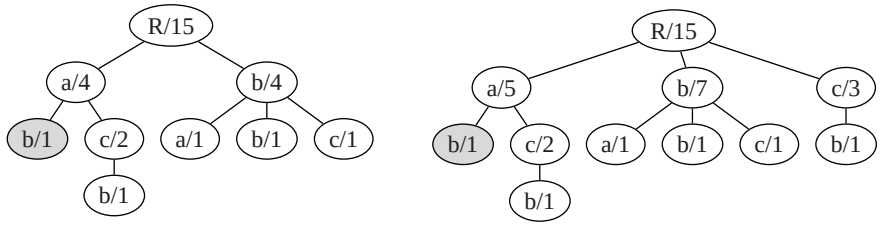
The maximum context that the $\mathcal{PPM}$ predictor can exploit is k; though, all intermediate contexts with length from 1 to $k-1$ can be used. This model is also referred as *all-Kth-Order* $\mathcal{PPM}$ model. The interleaving of various length contexts does not mean that this scheme is a variable length Markov predictor (although sometimes it is referred as such), because the decision on the context length is made beforehand and not in a data-driven way.

Apart from this basic scheme a number of variations have been proposed, which attempt to reduce the size of the trie by pruning some of its paths, based on statistical information derived from the input data. They set lower bounds for the normalized appearance count and for the conditional probabilities of subsequences and then prune any branch which does not exceed these bounds. Characteristic works adopting such an approach are reported in [10, 37, 16]. Apparently, these schemes are offline, making one or multiple passes over the input sequence in order to gather the required statistical information.

1.2.3.2 The Lempel-Ziv-78 scheme The Lempel-Ziv-78 Markov predictor, $\mathcal{LZ}78$ for short, is the second scheme whose virtues in carrying out predictions was investigated very early in the literature [56, 8]. The algorithm $\mathcal{LZ}78$ [64] arose from a need for finding a universal variable to fixed length coding method, and constructs a

prediction model for an input sequence as follows. It makes no assumptions about the maximal order for the generated model. Then, it parses the input sequence into a number of distinct subsequences, say $\mathfrak{s}_1, \mathfrak{s}_2, \ldots, \mathfrak{s}_x$, such that $\forall j \ (1 \leq j \leq x)$, the maximal prefix of subsequence $\mathfrak{s}_j$ is equal to some $\mathfrak{s}_i$, for some $1 \leq i < j$. The discovered subsequences, and the associated statistics are inserted into a trie in a manner identical to that of the $\mathcal{PPM}$ scheme. The computation of conditional probabilities takes place in a manner completely analogous to that of $\mathcal{PPM}$. The $\mathcal{LZ}78$ predictor for the sample sequence $aabacbbabbacbbc$ is depicted in the left part of Figure 1.2. Though, $\mathcal{LZ}78$ for this example is not able to produce a prediction for the test context ab (i.e., there is no subtree under the gray-shaded node).

Fig. 1.2 (Left) A $\mathcal{LZ}78$ Markov predictor for the sequence $aabacbbabbacbbc$. (Right) A $\mathcal{LZ}78$ predictor enhanced according to [8].



Apparently, the $\mathcal{LZ}78$ Markov predictor is an online scheme, it lacks administratively tuned parameters, like lower bounds on appearance counts, and it is a characteristic paradigm of a variable length Markov predictor. Although, strong results do exist which prove its asymptotic optimality and its superiority over any fixed length $\mathcal{PPM}$ predictor, in practice, various experimental studies contradict this result, because of the finite length of the input sequence. Nevertheless, the $\mathcal{LZ}78$ predictor remains a very popular prediction method. The original $\mathcal{LZ}78$ prediction scheme was enhanced in [8, 34] in a way such that apart from a considered subsequence which is going to be inserted into the trie, all its suffixes are inserted, as well (see right part of Figure 1.2).

1.2.3.3 The Probabilistic Suffix Tree scheme The Probabilistic Suffix Tree Markov predictor, $\mathcal{PST}$ for short, was introduced in [41], and it presents some similarities to $\mathcal{LZ}78$ and $\mathcal{PPM}$. Although, it specifies a maximum order for the contexts it will consider, it is actually a variable length Markov predictor and constructs its trie for an input sequence as follows. The construction procedure uses five administratively set parameters: k the maximum context length, P_{min} minimum normalized appearance count for any subsequence in order to be considered for insertion into the trie, r which is a simple measure of the difference between the prediction capability of the subsequence at hand and its direct father node, γ_{min} and α which together define the significance threshold for a conditional appearance of a symbol. Then, for every subsequence of length of 1 up to k, if it has never been encountered before, a new node is added to the trie labelled by this subsequence, provided that a set of three

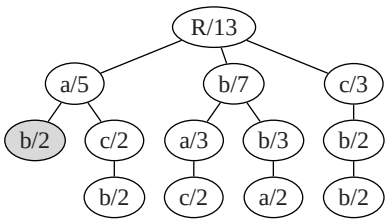
conditions hold. To exhibit the conditions, suppose that the subsequence at hand is $abcd$. Then, this subsequence will be inserted into the trie of the $\mathcal{PST}$ iff:

a) $E_n(abcd) \geq P_{min}$, and
b) There exists some symbol, say x, for which the following relations hold:

b$_1$) $\frac{E(abcdx)}{E(abcd)} \geq (1+a)\gamma_{min}$, and
b$_2$) $\frac{\widetilde{P}(x|abcd)}{\widetilde{P}(x|abc)} \geq r$ or $\leq 1/r \equiv \frac{E(abc)}{E(abcd)} * \frac{E(abcdx)}{E(abcx)} \geq r$ or $\leq 1/r$

The $\mathcal{PST}$ predictor with the following set of parameters $k = 3, P_{min} = \frac{2}{14}, r = 1.05, \gamma_{mim} = 0.001, \alpha = 0$ for the sample sequence $aabacbbabbacbbc$ is depicted in Figure 1.3. Apparently, $\mathcal{PST}$ is a subset of the baseline $\mathcal{PPM}$ scheme, when k is the same. $\mathcal{PST}$ for this example is not able to produce a prediction for the test context ab (i.e., there is no subtree under the gray-shaded node).

Fig. 1.3 A $\mathcal{PST}$ Markov predictor for the sequence $aabacbbabbacbbc$.

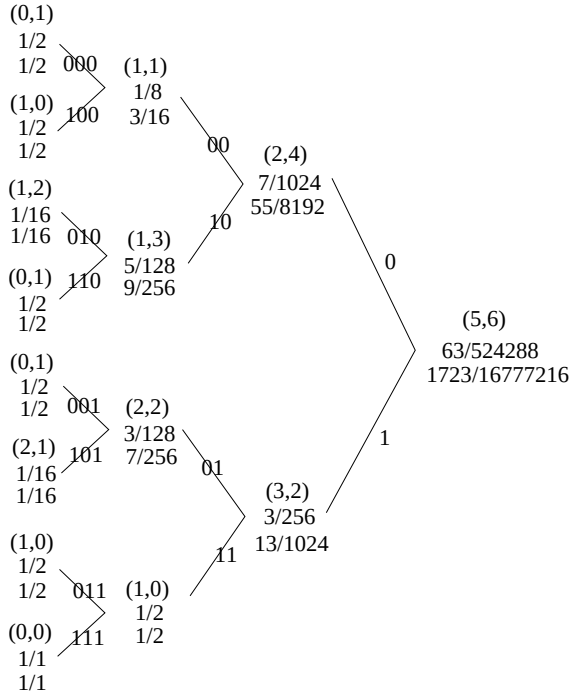


Apart from this basic scheme, a number of variations have been developed (e.g., [5], most of them providing improved algorithms, i.e., linear, for the procedures of "learning" the input sequence and for making predictions.

1.2.3.4 The Context Tree Weighting scheme The Context Tree Weighting Markov predictor [58], $\mathcal{CTW}$ for short, is based on the idea of combining exponentially many Markov chains of bounded order, and the original proposition dealt with binary alphabets only. The $\mathcal{CTW}$ assumes a pre-determined maximal order, say k, for the generated model and constructs a *complete binary tree* $\mathcal{T}$ of height k. The left and right children of a node s are denoted as $0s$ and $1s$, respectively. Each node s maintains two counters a_s and b_s, which count the number of zeros and ones, respectively, that followed context s in the input sequence so far. Additionally, each context (node) s maintains, apart from the pair (a_s, b_s), two probabilities P_e^s and P_w^s. The former, P_e^s, is the Krichevsky-Trofimov estimator for a sequence to have exactly a_s zeros and b_s ones. The latter probability, P_w^s, is the weighted sum of some values of P_e. The $\mathcal{CTW}$ predictor for the sample binary sequence $010|11010100011$ is depicted in the left part of Figure 1.4.

With P_e^R and P_w^R denoting the Krichevsky-Trofimov estimate and the $\mathcal{CTW}$ estimate of the root, respectively, we can predict the next symbol with the aid of a $\mathcal{CTW}$ as follows. We make the working hypothesis that the next symbol is a one,

Fig. 1.4 A $\mathcal{CTW}$ Markov predictor for the binary sequence $010|11010100011$.



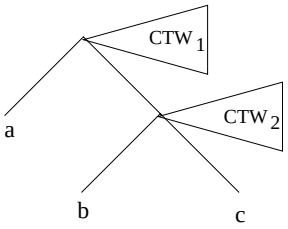
and we update the $\mathcal{T}$ accordingly obtaining a new estimate for the root $P_w^{'R}$. Then, the ratio $\frac{P_w^{'R}}{P_w^R}$ is the conditional probability that the next symbol is a one.

For the case of non-binary alphabets, Volf [57] proposed various extensions; among them, the *Decomposed* $\mathcal{CTW}$, $De\,\mathcal{CTW}$ for short, is the best compromise between method efficiency and simplicity. Firstly, we assume that the symbols belong to a alphabet Σ with cardinality $|\Sigma|$. We consider a full binary tree with $|\Sigma|$ leaves. Each leaf is uniquely associated with a symbol in Σ. Each internal node v defines the binary problem of predicting whether the next symbol is a leaf on v's left subtree or a leaf on v's right subtree. Then, we "attach" a binary $\mathcal{CTW}$ predictor to each internal node. We project the training sequence over the "relevant" symbols (i.e., corresponding to the subtree rooted by v) and translate the symbols on v's left (respectively, right) sub-tree to 0s (respectively, 1s). A diagram of the $De\,\mathcal{CTW}$ is depicted in Figure 1.5.

1.2.4 Comparison of prediction schemes

In the preceding section we described briefly the mechanics of four families of Markov predictors. In this section we will perform a qualitative comparison of the families; initially, we will comment on some generic features/advantages of the families and

Fig. 1.5 A sketch of the $De\,\mathcal{CTW}$ Markov predictor for the sequence $aabacbbabbacbbc$.



predictor	sequence it models
CTW_1	aa_a__a__a___
CTW_2	bcbbbbcbbc

then, we will elaborate on the kind of applications which could benefit from the prediction performance of each family.

Implicitly or explicitly all Markov predictors all based on the *short memory principle*, which says that the probability distribution of the next symbol can be approximated by observing no more than the last k symbols in that sequence. Some methods fix in advance the value of k (e.g., $\mathcal{PPM}$, $\mathcal{CTW}$). If the selected value for k is too low, then it will not capture all the dependencies between symbols, degrading its prediction efficiency. On the other hand, if the value of k is too large, then the model will overfit the training sequence. Therefore, variable length Markov predictors (e.g., $\mathcal{LZ}78$, $\mathcal{PST}$) are in general more appropriate from this point of view. This was the motivation for subsequent enhancements to $\mathcal{PPM}$ and $\mathcal{CTW}$ so as to consider unbounded length contexts, e.g., the $\mathcal{PPM}^\star$ algorithm [13].

On the other hand, variable length predictors face the problem of which sequences and of what length should be considered. $\mathcal{PST}$ attempts to estimate the predictive capability of each subsequence in order to store it in the trie, which results in deploying many tunable parameters. $\mathcal{LZ}78$ employs a prefix-based decorrelation process, which results in some recurrent structures to be excluded from the trie, at least at the first stages. This characteristic is not very important for infinite-length sequences, but may incur performance penalty for short sequences; for instance, the pattern bba is missing in both variants of $\mathcal{LZ}78$ of Figure 1.2. Although this example is by no means a kind of proof that $\mathcal{LZ}78$ is inferior to the other algorithms, it is an indication of how an individual algorithm's particularities may affect its prediction performance, especially in short sequences. Despite their superior prediction performance, $\mathcal{PPM}$ schemes are far less commonly applied than algorithms like $\mathcal{LZ}78$, which is favored over $\mathcal{PPM}$ algorithms for its relative efficiency in memory and computational complexity.

In Table 1.1 we summarize the Markov predictor families, their main members and their main qualitative characteristics.

From Table 1.1 we can gain some insights regarding which method is more appropriate for which type of application. Although, we emphasize that the choice and performance of a specific model largely depends on the application characteristics, it is the case that some results in the relevant literature show relative gains in the performance of one method w.r.t. the others for specific applications. To the best of our knowledge, we found no study which compares *all* families mentioned in this article for the location prediction issue with both synthetic and real data. In general,

Table 1.1 Qualitative comparison of discrete sequence prediction models.

Prediction method			Overheads			Particularity
Family	Variant	Markov class	Train	Parame/tion	Storage	
$\mathcal{LZ}78$	[8]	Variable	on-line	moderate	moderate	May miss patterns
	[64]	Variable	on-line	moderate	moderate	
$\mathcal{PPM}$	[10]	Fixed	off-line	heavy	large	Fixed length. High complexity
	[14]	Fixed	on-line	moderate	large	
	[16]	Fixed	off-line	heavy	large	
$\mathcal{PST}$	[5]	Variable	off-line	heavy	low	Parameterization
	[41]	Variable	off-line	heavy	low	
$\mathcal{CTW}$	[57]	Fixed	on-line	moderate	large	Binary nature
	[58]	Fixed	on-line	moderate	large	

the number of such studies is limited, and the real data they use (when they use such) come from limited settings (e.g., university campuses) and not from users of commercial wireless systems, e.g., cellular networks. Thus, it is not possible to draw safe conclusions from these works. Worthwhile studies containing comprehensive experiments with real data are reported in [37, 45, 11, 21].

Although alternative approaches could be possible, we prefer to present our suggestions for policy selection along two primary dimensions; the first dimension reflects the type of the problem (i.e., location prediction) and the second dimension reflects the "network part", where the prediction is carried out (i.e., fixed, resource-rich network servers or resource-starving mobile hosts).

For mobile applications, some very important intuitive results, which have also been confirmed experimentally[25], can be stated: a) user interests vary significantly with time (not "strong" stationarity), b) many alternative paths exist which lead to the same target location, thus the regularity patterns are "blurred" by noise. Due to the first observation, the possibility of using $\mathcal{LZ}78$ types of predictors is rather small. Due to the variance in the length of the individual client's trajectories, the rest of the variable-order Markov predictors are more appropriate; $\mathcal{PST}$ would be a perfect choice under the assumptions that the procedure is performed offline and it runs on a resource-rich server, or a relatively powerful laptop.

If energy conservation is the main issue in these applications (small portable devices like PDAs, mobile phones), then the choice of $\mathcal{PPM}$ style predictors seems more appropriate since they are online, but they sacrifice some prediction performance (due to the relatively small and fixed order model employed) for reduced model complexity. The second observation may turn all prediction methods inefficient, since it violates the "consecutiveness" property of appearance of the symbols in the patterns, upon which property all described Markov predictors rely. In cases where "noise symbols" are interleaved in pattern subsequences, the modified Markov predictors described in [37, 16] can be employed, but these algorithms are offline and require substantial resources (memory, power) to be executed. Therefore they

could only be used by fixed network servers, which collect huge amounts of user trajectories.

Location prediction is considered a relatively manageable problem, because of the few alternatives in possible contexts (i.e., hexagonal architecture of cellular systems, few fixed access points in wireless LANs and smart home applications) and because of the "strong" stationarity (i.e., few habitual routes in campuses/cities, few travel paths in urban regions – road network).

For location, prediction applications, several families of Markov predictors could be used in some specific scenarios each. For dynamic tracking of mobile hosts (with the tracking application running either in the network server or the mobile host) $\mathcal{PPM}$ and $\mathcal{LZ}78$ methods are appropriate. The small order $\mathcal{PPM}$ model and the enhanced $\mathcal{LZ}78$ [8] are expected to achieve the best performance, because of the undoubted validity of the stationarity assumption. Indeed, the study in [45] confirmed that intuitive results. These variants are perfect fit for dynamic resource allocation before handovers, as well. For location area design applications, where we are interested in discovering "long-standing" repetitive user routes, the process is offline and therefore methods like $\mathcal{PST}$ or [37, 16] are appropriate and less vulnerable to statistical deviation.

1.3 PREDICTIVE LOCATION INDEXING TECHNIQUES

In the previous section, we presented the dominant approaches for predictive location tracking for the case of a symbolic topology model. In many cases though, we are interested in tracking the location of mobiles at a finer granularity of space and time. Consider, for instance, our interest in tracking the trajectories of birds, airplanes or satellites, which are considered as points, or our interest in tracking the movement of a tropical storm, of fires. This interests stems from our need to answer queries like, "When two satellites are going to meet?", "Is the fire threatening village Thetidio?", and so on.

This leads to the idea of storing in a database for each moving object not the current position but rather a motion vector, which amounts to describing the position as a function of time. That is, if we record for an object its position at time t_0 together with its speed and direction at that time, we can derive expected positions for all times after t_0. Of course, motion vectors also need to be updated from time to time, but much less frequently than positions. Hence, from the location management perspective, we are interested in maintaining dynamically the locations of a set of currently moving objects and in being able to ask queries about the current positions, the positions in the near future, or any relationships that may develop between the moving entities and static geometries over time.

To support such functionality the database must build *indexes*; the main task if an index is to ensure fast access to single or several records in the database on the basis of a search key and thus avoid an otherwise necessary naive scan. Numerous indexes able to support continuous movement have appeared in the literature [18]. The indexes capable of accommodating moving objects can be generally categorized

into those optimizing queries about past states of movement and those tailored to serving queries about future positions of the moving objects; the first type of queries are called *historical*, while the second one are termed *predictive*. In the following, we will address the predictive location indexing techniques.

1.3.1 Assumptions and terminology

In the majority of the applications, the size and the shape of the moving objects are insignificant. Consequently, every object is modelled as a geometric point whose position consists a time function $\mathbf{x}(t)$ of the particular motion parameters. Based on $\mathbf{x}(\mathbf{t})$, the applications evaluate the future object locations. Furthermore, it is expected that the involved objects have the capacity and the obligation of periodically reporting any alteration occurring in $\mathbf{x}$ parameters. As an example, in most cases, $\mathbf{x}(t)$ is a linear function of time $\mathbf{x}(t) = \mathbf{x}(t_{\rm ref})+\mathbf{v}(t-t_{\rm ref})$, where the two parameters are the position $\mathbf{x}(t_{\rm ref})$ at the reference time $t_{\rm ref}$ and the velocity vector $\mathbf{v}$. Generally speaking, the equation parameters specify a dual to the time-location space framework.

As far as the type of queries is concerned, they can be classified as *range queries* and *proximity* or *nearest neighbor (NN)* ones. A range query is termed: (i) *time-slice* or *snapshot* (r, t) when, given a region r, usually a hyper-rectangle, located at time t, it asks for all moving objects contained in r at that time; (ii) *window* $(r, [t_1, t_2])$ when it asks for all objects crossing a hyper-rectangle r during a time interval $[t_1, t_2]$; (iii) *moving* $(r_1, r_2, [t_1, t_2])$ when it inquires all objects crossing the trapezoid formed by connecting the hyper-rectangle r_1 at time t_1 and the hyper-rectangle r_2 at time t_2; and (iv) *selectivity* or *aggregate* $(r, [t_1, t_2])$ when, given a region r and a time interval $[t_1, t_2]$, it requests an estimation about the number of the objects that will pass through r during $[t_1, t_2]$. Types (i)-(iii) are also known as *range reporting* whereas type (iv) is well tailored to the case of limited memory and strict real-time processing demand.

On the other hand, a proximity query inquires for the k nearest moving objects to a given location at a time instance t or during a time interval $[t_1, t_2]$, with $k \geq 1$. Sometimes, all objects having a given location as their nearest neighbor are asked for; this kind of searching is termed *reversed nearest neighbor searching (RNN)*.

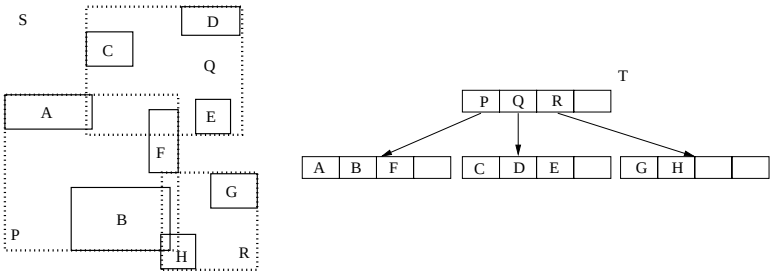
In the above definitions, two variations are possible: Firstly, the query range/point also moves which makes the query *continuous*. And secondly, the answer set may be returned with temporal or spatial validity information, informing thus the user about the future time t the result expires or the validity region r containing the query position within which the answer remains valid.

Concluding this subsection, we briefly discuss the R-tree [19], a general purpose practical indexing structure, since the majority of the solutions to be presented are built upon it. So, an R-tree is a height-balanced tree which can be considered as an extension of the B^+-tree for multi-dimensional data. The minimum bounding rectangle (MBR) of each geometric object, along with a pointer to the disk address where the object actually resides, are stored into the leaves. Each internal node entry consists of a pair (pointer to a subtree $\mathcal{T}$, MBR of $\mathcal{T}$), with the MBR of a tree $\mathcal{T}$ defined as the MBR enclosing all the MBRs stored in $\mathcal{T}$. Like in B^+-trees, each node contains at least m and at most M entries, where $m \geq M/2$. On the other

hand, unlike B^+-trees, a search query may activate several search paths from the root to the R-tree leaves, resulting, in the worst case, in a linear to the size of data set performance just to retrieve a few objects.

Figure 1.6 illustrates an R-tree instance on a set S of rectangles. Since its introduction, several variants of the R-tree have been proposed, each one aiming at improving the performance by tuning some parameters. Among the members of the "R-tree family", the most prominent one is the R^*-tree of Beckmann et al. [6].

Fig. 1.6 An R-tree instance: (a) A set S of rectangles A–H, (b) The corresponding R-tree T



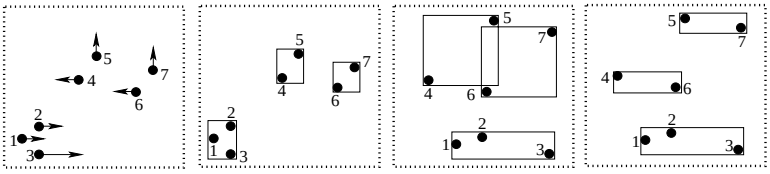
1.3.2 Indexing structures for range queries

1.3.2.1 Range reporting. Tayeb et al. [55] presented one of the first works on indexing mobile objects for snapshot and window queries, by reducing the problem to indexing lines in a periodically rebuilded bucket PR Quadtree [44] with high space requirements. Subsequently, Kollios et al. [27] presented solutions for one- and two-dimensional range searching which index objects either in xt-plane using R-trees [19] or in the two-dimensional dual space deploying dynamic external partition trees [2]; the last proposal is mainly of theoretical interest due to the employed index structures.

In [43] a time-parameterized version of $R^\star$-trees [6], called TPR-tree, was introduced, because none of the memebers of the R-tree family of indexes is efficient in indexing mobile objects. Let us explain this with the aid of Figure 1.7. The leftmost digram shows the positions and velocity vectors of 7 point objects at time 0. Assume we create an R-tree at time 0. The second diagram shows one possible assignment of the objects to minimum bounding rectangles (MBRs) assuming a maximum of three objects per node. Previous work has shown that attempting to minimize the quantities known as overlap, dead space, and perimeter leads to an index with good query performance [6], and so the chosen assignment appears to be well chosen. However, although it is good for queries at the present time, the movement of the objects may adversely affect this assignment. The third diagram shows the locations of the objects and the MBRs at time 3, assuming that MBRs grow to stay valid. The grown MBRs adversely affect query performance; and as time increases, the MBRs will continue to grow, leading to further deterioration. Even though the objects be-

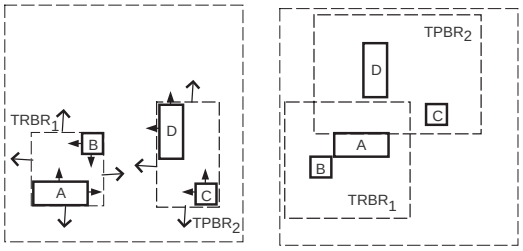
longing to the same MBR (e.g., objects 4 and 5) were originally close, the different directions of their movement cause their positions to diverge rapidly and hence the MBRs to grow. From the perspective of queries at time 3, it would have been better to assign objects to MBRs as illustrated by the rightmost diagram. Note that at time 0, this assignment will yield worse query performance than the original assignment. Thus, the assignment of objects to MBRs must take into consideration when most queries will arrive.

Fig. 1.7 Mobile objects and resulting leaf-level MBRs.



The TPR-tree is capable of accommodating moving objects with constant velocities in one-, two- and three-dimensional space, and is the de facto spatial index for future queries. The MBRs in this example illustrate the kind of time-parameterized bounding rectangles supported by the TPRtree. The novelty of the approach are the time-parameterized bounding rectangles (TPBRs) being employed, associated with a velocity vector (see Figure 1.8): for each coordinate x_i, the lower bound is set to be the minimum observed x_i-coordinate value at time t_{ref}, moving with the minimum observed velocity, and the upper bound is defined to be the maximum x_i-coordinate value at time t_{ref}, moving with the maximum observed velocity. Since the TPBRs never shrink and are conservatively bounded, the index is tuned for H time units; after that, a global rebuilding of the structure is conducted. TPBRs also impose the generalization of the update and rebuilding algorithms of the R^*-trees so that their respective objective functions are time parameterized. In summary, TPR-trees support all types of range queries (time-slice queries, window queries and moving) while proven to be a very practical solution.

Fig. 1.8 An example of forming Time-Parameterized Bounding Rectangles.



Agarwal et al. [3] provided solutions of theoretical interest for answering window and moving window queries for one- and two-dimensional moving points with time

complexities that depend on the number of events — i.e., alteration of the relative w.r.t. an axis location order between objects or insert/deletion of a moving object— that occurred between the current time and the query time. Their approach is based on a careful segmentation of the plane/space into a logarithmic number of strips (slabs), each one containing a bounded number of events. The arrangements of every slab are then stored in persistent versions of B-trees. As a result, moving window queries have a time complexity of $O(\log_B n + k/B + B^{i-1})$ and $O(\sqrt{n/B^i}(B^{i-1} + \log_B n) + k/B)$, for one- and two-dimensional movements, respectively, B being the page capacity and i the slab number.

R^{EXP}-trees [42] use also TPBRs for bounding moving objects. However, capitalizing on the assumption that objects expire after a time period $t_{\exp}$, within which they have not reported their position, the author derived analytical formulas which produce tighter TPBRs. This fact in conjunction with the lazy removal of expired objects only after an update operation discovers an expired entry make R^{EXP}-trees to exhibit better experimental performance with respect to TPR-trees. STAR-trees [39] were also introduced as an improvement over TPR-trees for the two-dimensional case. The main feature of this proposal is self-adjustment: based upon user specifications concerning space overhead and performance quality, the index self-adjusts without any user interference. Towards this end, the extends of the points are constantly approximated, with the aid of a priority queue, and the children of a node are redistributed whenever they overlap too much. The authors reported the following improvements: a speed-up of 2.3 with respect to TPR-tree was achieved and the deterioration of the scheme over time is quite small.

TPR^*-trees [52] improved TPR-trees by introducing more elaborated decision making processes for insertion path selection, node re-insertion and children node redistribution which pay-off a lot in terms of search performance. Additionally, the authors suggested a cost model for the original TPR-tree which emphasizes the factors that influence its performance and shows the superiority of TPR^*-tree over TPR-trees. This fact was also confirmed by an extensive experimental investigation which demonstrated that with the TPR^*-tree the average query cost is almost five times less, whereas the average update cost is nearly constant.

In [1] three theoretical indexing schemes for moving points in the plane were introduced. The first one improves upon the approach of [27] by using a two-level external partition tree and has time complexity of $O(n^{1/2+\epsilon} + k)$, k the output size. The other two support one- and two-dimensional queries referring to the present time or arriving in a strictly chronological order and have logarithmic complexity. The one-dimensional index employs kinetic B-trees while the two-dimensional kinetic range trees.

Patel et al. [38] introduced the STRIPE index, which basically is a multidimensional external PR bucket quadtree supporting all three types of range queries (time slice, window and moving). Each moving object is represented in dual $2d$-dimensional parametric space. The parametric space is indexed by applying a disjoint regular partitioning mainly imposed by the underlying quadtree. The authors tested the performance of STRIPE against the TPR^*-tree thoroughly and found that their proposal is faster in both update and query time; namely, queries are faster by a

factor of four while the update operations complete by an order of magnitude quicker since TPR*-trees have the disadvantages of poor cache locality and multiple path traversals.

[48] coped with the case of circular static range query during a time interval for moving objects with unknown moving patterns. In order to support arbitrarily movements, Tao et al. suggested a monitor-and-index framework: Each moving object individually and constantly computes the recursive function that best describes its movement using motion matrices introduced by the authors. On the other hand, the server adopts the same coarse polynomial function m for every object. The latter fact introduces imprecision which is dealt with a two step process. During the filtering phase, all objects that either definitely or possibly satisfy the query are determined using STP-tree. STP-tree can be considered as a generalization of both TPR and TPR^*-trees to arbitrary polynomial functions m. In a second refinement step, the server communicates with the doubtful objects which evaluate the query according to their own accurate moving function and send back the result if necessary along with the corrected parameters of m. The applicability of the proposal is thoroughly investigated through a series of experiments that concern both the movement approximation method and the new index.

1.3.2.2 Range aggregate. In 2002, Choi and Chung [12] presented the first work on rectangular range selectivity for static queries. Their solution for the one-dimensional case is based on the simple observation that a point satisfies the query range r if and only if the segment formed by its (tentative) position at the start and the end of the query time interval intersects r. The authors proposed a histogram based counting solution so that the spatial universe is partitioned into a number of time evolving buckets. This method was also extended to the two-dimensional case by projecting objects and queries on each spatial dimension and evaluating the selectivity as the product of the one-dimensional results.

The previous method suffers from overestimated results since the projection overlooks necessary temporal conditions. Tao et al. [54] achieved better estimation results by dropping the projection step; their solution tackles directly the problem with the deployment of a spatio-temporal histogram which considers both locations and velocities. The authors report significant improvements in selectivity precision and manipulation of updates. In [20] one- and two-dimensional movements were also coped with. As a matter of fact, two solutions were introduced: One is also based on multidimensional histograms which, none the less, are defined on dual space. The accommodation of the moving objects into an index is prerequisite for the second solution. Namely, the summaries of each leaf entry, in the form of number of objects and their bounded rectangular spatial extent, are stored in a hash table which is used for output extrapolation.

In contrast to the previous three solutions, in [53] random sampling was chosen for selectivity estimation in order to spare space and processing overhead. Particularly, the introduced method of Venn sampling employs a set S of m pivot queries which represent the actual distribution and are perfectly estimated. The most interesting part is that, based on S, a weighted sample of m moving objects, non necessarily

belonging to the underlying data set, is formed and communicated to the moving objects. This sample is queried against any incoming query and the sum of weights of qualifying objects is returned to the user. The system constantly monitors the quality of the sample set in collaboration with the moving objects; in case the estimation error surpluses a threshold, the sample weights but not the respective sample objects are adjusted. The authors present experimental results rendering the proposal quite appealing.

1.3.3 Indexing structures for nearest neighbor queries

Kollios et al. [26] introduced practical yet preliminary solutions for locating the nearest moving neighbor of a static query in the plane permitting either arbitrarily or restricted in fixed line segments object movement. They consider both the cases of indexing in the xt-plane and in the dual plane. The first case allows indexing with standard spatial indexes, like, for example, R-trees [19] while the second one utilizes B^+-trees [15] and dual plane segmentation in horizontal stripes.

Song and Roussopoulos [46] studied the continuous kNN problem, which inquires for the k nearest static neighbors of a moving query point. Their approach is based on the observation that as the query point is moving to new locations, some previously reported neighbors are still among the k closest ones. This fact is formally expressed by a series of conditions which allow the employment of standard indexing structures like R-trees.

In [63] Zheng and Lee considered enhancing NN queries with validity information. The approach is quite simple: The nearest neighbor of a moving point is calculated using the Voronoi diagram of the static data set. Since the Voronoi cell c of the nearest neighbor is available, the maximum circle around the query point which does not cross any boundary edge of c consists a safe lower bound for the validity of the query result.

In [7] two-dimensional nearest neighbor and reverse nearest neighbor queries for a query point q during a time interval t were treated by properly extending TPR-tree algorithms. NN queries apply a depth-first search technique which constantly prunes TPBRs with no chance enclosing closest points. The RNN queries are more elaborate: the space around the query point q is divided into six equal sectors s_i by straight lines intersected at q, since there must exist at most six RNN points, at most one in each s_i. Therefore, sectors containing two or more nearest points of q are discarded and the search is restricted among the nearest neighbors of the rest.

Continuous NN queries for static data sets were also treated, albeit from a different perspective, in [22]. This solution is built upon ellipsoid areas around the moving query point which are generated utilizing current, past and future trajectory positions and carefully selected metrics. The static data set, on the other hand, is indexed with a "spatial" structure, like, for example, R-trees. Since the previous answers are not reused as the query point changes position, this method needs involved tuning to be effective.

Tao et al. [51] considered continuous kNN search when the static input data set is accommodated in an R-tree T and the query point is linearly moving. When $k = 1$,

these assumptions guarantee that the output set consists a sequence of points p_i, which partition the movement line segment into a sequence of disjoint segments s_i. Every point in s_i has p_i as its nearest neighbor. These facts suggest a branch-and-bound investigation of T employing heuristic node pruning. Tao et al. also generalized the pruning rules for the continuous kNN case, providing an extensive experimental evaluation of their proposal.

Aggarwal and Agrawal [4] introduced NN solutions for objects moving in nonlinear trajectories of arbitrary dimensionality whose parametric representation satisfy the so called convex hull property. d-dimensional trajectories with constant velocity, d-dimensional parabolic trajectories, elliptic orbits, and trajectories that accept approximate Tailor expansion belong to this category. Due to the convex hull property, the locality in parametric space corresponds to the locality in the positions of objects, and, thus NN search can be accomplished by a branch-and-bound, best-first algorithm conducted in a classical spatial index. The authors demonstrated their method for linear and parabolic trajectories in three- and two-dimensional space respectively.

TPR-trees algorithms were enhanced in [40], so that continuous kNN queries on moving points during a time interval $[t_1, t_2]$ can be served. The suggested solution capitalizes on the following geometric fact: the kNN points can be determined by the k levels of the arrangement of the squared distance functions of the moving points with respect to the moving query during $[t_1, t_2]$. Therefore, after collecting at least k points by a depth-first traversal of the underlining TPR-tree according to the minimum squared distance metric between the moving query and the bounding rectangles at t_1, the kNN points of $[t_1, t_2]$ are determined. Then, in a second stage, the TPR-tree is once more traversed, in order to refine and finalize the output set.

Iwerks et al. [23] also presented algorithms for answering continuous kNN queries on a constantly moving point set with respect to either a static or a moving query point. Their approach runs also in two phases. During first phase, input set is filtered with a continuous query which asks about objects within a distance bound d. Then, the qualified points are ranked with a priority queue, which tracks the time instances when points change their distance to the query point or when points change their order with respect to a current kNN point.

Nearest neighbor queries, among others, were also treated in [1]. Specifically, Agarwal et al. suggested an algorithm which offers approximate results to NN searching by replacing the Euclidean metric with a polyhedral one. The input set is accommodated into a three level composite index of $O(n^{1/2+\epsilon}/\sqrt{\delta})$ time complexity, $0 < \delta, \epsilon < 1$. The first two levels are external partition trees on dual space. The last level stores the lower envelope of the trajectories in linear lists.

The Conceptual Partitioning method (CPM) was introduced in [35] for constantly monitoring of multiple continuous NN queries in highly dynamic environments. In brief, the space is partitioned by a regular grid and indexed in main memory. Every cell of this grid maintains the list of objects residing therein and every posed query along with its current result set is stored in a table. Additionally, CPM imposes a total order on the cells around every query based on proximity criterion. In that way, every update in both the data and the query set is tackled with minimal computational

costs without any assumptions about the occurring moving patterns; this is depicted by both qualitative and thoroughly experimental analysis.

Afterwards the previous work, Mouratidis et al. [36] presented a main memory solution for incremental monitoring of continuous kNN queries when the query and the data objects move in road networks. Their approach basically is based on network expansion about the query until k nearest neighbors are collected. The formed shortest path tree is stored along with the query so that any updates are smoothly incorporated. Moreover, the authors proposed a method for computation sharing among queries whose shortest paths are crossed. All suggested solutions are experimentally evaluated.

Finally, [30] treated continuous nearest surrounder (NS) queries which asks for the nearest neighbors at individual distinct angles from the query point. NS queries, thus, monitor the nearest neighbors around the queries by considering both the distance and the angular attributes. The system registers the objects' locations in an R-tree and the queries, along with their current results, in a hash table so that any update of either data or query objects can be incrementally evaluated, capitalizing on the notion of 'safe regions' firstly introduced in [29].

1.3.4 Indexing structures for both window and nearest neighbor queries

In [50] time-parameterized window and kNN queries (TP) were introduced which, along with the objects that satisfy the spatial conditions, also return the expiration time of the validity of the answer and the change that invalidates the answer at that time. The key concept of this approach is the influence time which is associated with every moving object o and indicates the time o influences the validity of the answer. By definition, the expiration time of the answer equals to the minimum influence time of all objects which, in turn, can be evaluated with a NN search where the distance metric is the influence time. This observation is valid for both window and kNN queries. The authors proved analytical formulas for evaluating the influence times of an object. As a result, standard branch-and-bound traversal of the index accommodating the object set can be employed. Their solution also treats continuous spatio-temporal queries by posing a TP query every time the current result expires. Finally, TP queries can also serve earliest event queries which ask for the earliest time in the future a certain event could take place; for example, one may need to figure out the first time a moving query point q meets another moving object. By surrounding q with a time-varying radius cycle, this query reduces to TP by evaluating the earliest time the circle contains a point, which, in turn, equals to determining the smallest radius of such a circle.

Building upon [50], Zhang et al. [61] dealt also with validity kNN and window queries. In the first case, order-k Voronoi cells comprise the validity regions. These can be found implicitly by, firstly, generating the nearest neighbors and then issuing time-parameterized kNN queries for locating which points define their border. In the second case, the maximal rectangle r around the center of the window, within which the result remains unchanged, is firstly evaluated. Then, r is refined by subtracting the parts that would force the query to mistakenly contain points not in the reported

answer. These two steps involve one standard window query, one "holey" window query and few main memory TP window queries.

Tao et al. [49] proved a number of theoretical bounds on validity range and NN queries. Specifically, when the query's length and movement are chosen from a constant number of combinations and the point set is static, the query cost is logarithmic and the space is linear. When the point set is static, the query length is arbitrarily, and the movement is axis-parallel, then the time complexity is $O(\log_B^2(n/B)/\log_B\log_B(n/B))$ and the space cost is $O(n/B\log_B(n/B)/\log_B\log_B(n/B))$. On the other hand, in the case of static input point set and queries with arbitrary length and movement the space complexity is $O(n/B)$ while the query costs $O((n/B)^{1/2+\epsilon})$. When the data points are dynamic and the query is static, the query has logarithmic complexity and space is bounded by $O(n^2/B\log_B(n/B))$. The case of both dynamic data points and query point is only considered in one-dimensional space, proving a linear space and logarithmic time complexity. As far as the NN search query is concerned, when the input point set is static lying in the plane or comprised of moving points on the line, the solution is of linear space and logarithmic query cost.

The B^x-trees were introduced in [24], that are capable of serving range, and kNN queries as well as their continuous counterparts. The main ingredient of this method is movement linearization: The time axis is partitioned into intervals of Δ time units, and each interval is further subdivided into n subintervals of equal length. Every object, according to its t_{ref}, is assigned to one subinterval. Within each subinterval, the positions of the objects fallen in are linearized according to a space filling curve and then stored into a B^+-tree. Therefore, the B^x-tree is actually a sequence of B^+-trees evolving as time goes by. The authors conducted extensive experiments which show the superiority of B^x-trees over TPR-trees for all kinds of queries. Here we must note that BB^x-trees [32] were introduced as a natural extension of B^x ones, able of answering both predictive and historical queries.

After observing that the linearization process of [24], in [59] only object's locations were considered, leading thus to excessive false hits, proposed B^{dual}-trees. B^{dual}-trees also deploy B^+-trees indexing, non the less, a space filling curve which is based on both object locations and velocities. The authors demonstrated the superiority of their method over B^x-trees both analytically (that is, using derived formulas) and experimentally—the experimental study also gives data for STRIPES and TPR^*-trees. In a nutshell, B^{dual}-trees can be considered as the state of the art solution in its category.

1.3.5 Evaluation of predictive indexing

In overall, TPR-trees [43] and its descendant variations, like TPR^*-trees [52], can be considered the most appropriate and practical choice for serving range queries. In case of limited resources, the Venn sampling technique of [53] is deemed a natural option. As far as nearest neighbor queries is concerned, the TPR-tree variation of [40] and the conceptual partitioning method (CPM) of [35] are both competent

candidates. Finally, B^x-trees [24] and B^{dual}-trees [59] proved to be apt solutions when one wants to treat equally well range and nearest neighbor queries.

Regarding future steps toward developing more indexing mobiles, it would be very helpful the indexes to provide results that capture the uncertainty associated to the location of moving objects due to network delays and the continuous character of motion. It would be also very interesting to efficiently cope with non-linear trajectories since the scope and the range of indexable moving objects will be significantly extended. Another appealing subject, especially for extending mobile application capabilities, is the design of indexing structures capable of serving mixed queries concerning the past and the future of movement The incremental valuation of validity queries is very intriguing, as well. Finally, from an engineering perspective, it would be very helpful: (i) to test all indexes with real datasets, as, until now, every experimental investigation is conducted with semi-real ones, where the movement component is actually generated; and (ii) to design efficient updating algorithms for the indexes, different from the usual 'deletion and re-insertion' practice, accepting perhaps a tradeoff between either the query time or the accuracy of the result and the update time.

1.4 CONCLUSION

This chapter identified the inherent uncertainty in the movement of mobiles in areas covered by wireless networks, and the problems caused to resource allocation due to this uncertainty. Starting from this fundamental observation, it subsequently recognized the benefits of being able to forecast the future locations of the mobile hosts. This ability could be used to act proactively, instead of reactively, to many situations. For instance, having estimates of future positions of mobile hosts, the network could take appropriate decisions regarding the bandwidth that will allocate to the cells containing these locations. In addition, in wireless ad hoc networks, where communication between nodes is performed on a store-and-forward basis for nodes not in close proximity, the communication could be deferred until the nodes come closer to each other, thus saving network resources, like precious bandwidth and storage space in the intermediate nodes, reducing packet collisions and so on.

Though, predictive location tracking can be performed only if the mobiles' movements exhibit some degree of regularity, thus making the construction of mobility models feasible. The generic principle governing location prediction can be summarized in a short sentence: *study the present and project to the future*. Exploiting this principle, the issue of location prediction turned out to be a matter of recording the present mobile trajectories, and developing mobility models from these. The storage of the trajectories should be of the kind to allow compact representation and at the same time efficient generation of predictions.

Subsequently, we investigated two different scenarios for predictive location prediction. According to the first scenario, the roaming area of the mobiles can be considered as a union of non-overlapping cells of arbitrary geometry, and according to the second scenario, the location tracking is performed at the granularity of geo-

graphical coordinates. For the first scenario, we modelled the problem of predictive location tracking in terms of the discrete sequence prediction problem. For this latter problem, we presented Markov predictors as a practical and high performance solution, categorized them into four families giving their qualitative characteristics, their strengths and their weaknesses. For the second scenario, which is more tightly coupled with location databases, we surveyed the state-of-the-art techniques for constructing indexes capable of answering queries which mainly concern various complex future predicates.

Undoubtedly, predictive location tracking is very important for reducing the latency and the resource consumption in any wireless network or for prolonging the lifetime of wireless ad hoc networks. Though, the problem is not easily manageable due to the difficulty of constructing models representing the actual mobile trajectories. Although very significant steps have been taken towards achieving this target, work is still needed to characterize the predictability of mobile trajectories, to analyze collections of real mobile trajectories, to develop more effective prediction models and mainly to develop distributed models of prediction through cooperation, which will be suitable for the emerged area of wireless sensor networks.

Acknowledgments

Research supported by a ΓΓΕΤ grant in the context of the project "Data Management in Mobile Ad Hoc Networks" funded by ΠΥΘΑΓΟΡΑΣ II national research program.

REFERENCES

1. P. K. Agarwal, L. Arge, and F. Erickson. Indexing moving points. *Journal of Computer and System Sciences*, 66(1):207–243, 2003.

2. P. K. Agarwal, L. Arge, F. Erickson, P. G. Franciosa, and J. S. Vitter. Efficient searching with linear constraints. *Journal of Computer and System Sciences*, 61(2):194–216, 2000.

3. P. K. Agarwal, L. Arge, and J. Vahrenhold. Time responsive external data structures for moving points. In *Proceedings of the International Workshop on Distributed Algorithms and Data Structures (WADS)*, volume 2125 of *Lecture Notes in Computer Science*, pages 50–61, 2001.

4. C. C. Aggarwal and D. Agrawal. On nearest neighbor indexing of nonlinear trajectories. In *Proceedings of the ACM Symposium on Principles Of Database Systems (PODS)*, pages 252–259, 2003.

5. A. Apostolico and G. Bejerano. Optimal amnesic probabilistic automata or how to learn and classify proteins in linear time and space. *Journal of Computational Biology*, 7(3–4):381–393, 2000.

6. N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R^*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 322–331, 1990.

7. R. Benetis, C. S. Jensen, G. Karciauskas, and S. Saltenis. Nearest neighbor and reverse nearest neighbor queries for moving objects. *Very Large Data Bases Journal*, 15(3):229–250, 2006.

8. A. Bhattacharya and S. K. Das. LeZi-Update: An information-theoretic framework for personal mobility tracking in PCS networks. *ACM/Kluwer Wireless Networks*, 8(2–3):121–135, 2002.

9. S. Chakraborty, Y. Dong, D. K. Y. Yau, and J. C. S. Lui. On the effectiveness of movement prediction to reduce energy consumption in wireless communication. *IEEE Transactions on Mobile Computing*, 5(2):157–169, 2006.

10. X. Chen and X. Zhang. A popularity-based prediction model for Web prefetching. *IEEE Computer*, 36(3):63–70, 2003.

11. F. Chinchilla, M. Lindsey, and M. Papadopouli. Analysis of wireless information locality and association patterns in a campus. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, volume 2, pages 906–917, 2004.

12. Y.-J. Choi and C.-W. Chung. Selectivity estimation for spatio-temporal queries to moving objects. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 440–451, 2002.

13. J. G. Cleary and W. J. Teahan. Unbounded length contexts for PPM. *The Computer Journal*, 40(2–3):67–75, 1997.

14. J. G. Cleary and I. H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32(4):396–402, 1984.

15. D. Comer. The ubiquitous B-tree. *ACM Computing Surveys*, 11(2):121–137, 1979.

16. M. Deshpande and G. Karypis. Selective Markov models for predicting Web page accesses. *ACM Transactions on Internet Technology*, 4(2):163–184, 2004.

17. M. Feder, N. Merhav, and M. Gutman. Universal prediction of individual sequences. *IEEE Transactions on Information Theory*, 38(4):1258–1270, 1992.

18. R. H. Güting and M. Schneider. *Moving Objects Databases*. Series in Data Management Systems. Morgan-Kaufmann, 2005.

19. A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 47–57, 1984.

20. M. Hadjieleftheriou, G. Kollios, and V. J. Tsotras. Performance evaluation of spatio-temporal selectivity estimation techniques. In *Proceedings of the IEEE International Conference on Statistical and Scientific Database Management (SSDBM)*, pages 202–211, 2003.

21. M. Halvey, M. Keane, and B. Smyth. Mobile Web surfing is the same as Web surfing. *Communications of the ACM*, 49(3):76–81, 2006.

22. Y. Ishikawa, H. Kitagawa, and T. Kawashima. Continual neighborhood tracking for moving objects using adaptive distances. In *Proceedings of the IEEE International Database Engineering and Applications Symposium (IDEAS)*, pages 54–63, 2002.

23. G. S. Iwerks, H. Samet, and K. Smith. Continuous k-nearest neighbor queries for continuously moving points with updates. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 512–523, 2003.

24. C. S. Jensen, D. Lin, and B. C. Ooi. Query and update efficient B^+-tree based indexing of moving objects. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 768–779, 2004.

25. D. Katsaros and Y. Manolopoulos. Prediction in wireless networks by Markov chains. *IEEE Wireless Communications magazine*, 2007. to appear.

26. G. Kollios, D. Gunopoulos, and V. J. Tsotras. Nearest neighbor queries in a mobile environment. In *Proceedings of the International Workshop on Spatio-Temporal Database Management (STDBM)*, volume 1678 of *Lecture Notes in Computer Science*, pages 119–134, 1999.

27. G. Kollios, D. Gunopoulos, and V. J. Tsotras. On indexing mobile objects. In *Proceedings of the ACM Symposium on Principles Of Database Systems (PODS)*, pages 261–272, 1999.

28. M. Kyriakakos, N. Frangiadakis, L. Merakos, and S. Hadjiefthymiades. Enhanced path prediction for network resource management in wireless LANs. *IEEE Wireless Communications*, 10(6):62–69, 2003.

29. K. C. K. Lee, W.-C. Lee, and H. V. Leong. Nearest surrounder queries. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, 2006.

30. K. C. K. Lee, J. Schiffman, W.-C. Zheng, B. Lee, and H. V. Leong. Tracking nearest surrounders in moving object environments. In *Proceedings of the IEEE International Conference on Pervasive Services (ICPS)*, pages 3–12, 2006.

31. D. A. Levine, I. F. Akyildiz, and M. Naghshineh. A resource estimation and call admission algorithm for wireless multimedia networks using the shadow cluster concept. *IEEE/ACM Transactions on Networking*, 5(1):1–12, 1997.

32. D. Lin, C. S. Jensen, B. C. Ooi, and S. Saltenis. Efficient indexing of the historical, present, and future positions of moving objects. In *Proceedings of the IEEE International Conference on Mobile Data Management (MDM)*, pages 59–66, 2005.

33. T. Liu, P. Bahl, and I. Chlamtac. Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks. *IEEE Journal on Selected Areas in Communications*, 16(6):922–936, 1998.

34. A. Misra, A. Roy, and S. K. Das. An information-theoretic framework for optimal location tracking in multi-system $4G$ wireless networks. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, volume 1, pages 286–297, 2004.

35. K. Mouratidis, M. Hadjieleftheriou, and D. Papadias. Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 634–645, 2005.

36. K. Mouratidis, M. L. Yiu, D. Papadias, and N. Mamoulis. Continuous nearest neighbor monitoring in road networks. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 43–54, 2006.

37. A. Nanopoulos, D. Katsaros, and Y. Manolopoulos. A data mining algorithm for generalized Web prefetching. *IEEE Transactions on Knowledge and Data Engineering*, 15(5):1155–1169, 2003.

38. J. M. Patel, Y. Chen, and V. P. Chakka. STRIPES: An efficient index for predicted trajectories. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 637–646, 2004.

39. C. M. Procopiuc, P. K. Agarwal, and S. Har-Peled. $STAR$-tree: An efficient self-adjusting index for moving objects. In *Proceedings of the Workshop on Algorithm Engineering and Experiments (ALENEX)*, volume 2409 of *Lecture Notes in Computer Science*, pages 178–193, 2002.

40. K. Raptopoulou, A. Papadopoulos, and Y. Manolopoulos. Fast nearest-neighbor query processing in moving-objects databases. *GeoInformatica*, 7(2):113–137, 2003.

41. D. Ron, Y. Singer, and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25(2–3):117–149, 1996.

42. S. Saltenis and C. S. Jensen. Indexing of moving objects for location-based services. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, 2002. 463–472.

43. S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez. Indexing the positions of continuously moving objects. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 331–342, 2000.

44. H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, MA, 1990.

45. L. Song, D Kotz, R. Jain, and X. He. Evaluating location predictors with extensive Wi-Fi mobility data. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, volume 2, pages 1414–1424, 2004.

46. Z. Song and N. Roussopoulos. k-nearest neighbor for moving query point. In *Proceedings of the International Symposium on Advances in Spatial and Temporal Databases (SSTD)*, volume 2121 of *Lecture Notes in Computer Science*, pages 79–96, 2001.

47. S. Tabbane. Location management methods for third-generation mobile systems. *IEEE Communications magazine*, 35(8):72–78 & 83–84, 1997.

48. Y. Tao, C. Faloutsos, D. Papadias, and B. Liu. Prediction and indexing of moving objects with unknown motion patterns. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 611–622, 2004.

49. Y. Tao, N. Mamoulis, and D Papadias. Validity information retrieval for spatio-temporal queries: Theoretical performance bounds. In *Proceedings of the International Symposium on Advances in Spatial and Temporal Databases (SSTD)*, volume 2750 of *Lecture Notes in Computer Science*, pages 159–178, 2003.

50. Y. Tao and D. Papadias. Time-parameterized queries in spatio-temporal databases. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 334–345, 2002.

51. Y. Tao, D. Papadias, and Q. Shen. Continuous nearest neighbor search. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 287–298, 2002.

52. Y. Tao, D. Papadias, and Q. Sun. The TPR^*-tree: An optimized spatio-temporal access method for predictive queries. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 790–801, 2003.

53. Y. Tao, D. Papadias, J. Zhai, and Q. Li. Venn sampling: A novel prediction technique for moving objects. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 680–691, 2005.

54. Y. Tao, J. Sun, and D. Papadias. Selectivity estimation for predictive spatio-temporal queries. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 417–428, 2003.

55. J. Tayeb, O. Ulusoy, and O. Wolfson. A quadtree-based dynamic attribute indexing method. *The Computer Journal*, 41(3):185–200, 1998.

56. J. S. Vitter and P. Krishnan. Optimal prefetching via data compression. *Journal of the ACM*, 43(5):771–793, 1996.

57. P. Volf. *Weighting techniques in data compression: Theory and algorithms*. PhD thesis, Technische Universiteit Eindhoven, 2002.

58. F. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens. The context-tree weighting method: Basic properties. *IEEE Transactions on Information Theory*, 41(3):653–664, 1995.

59. M. Yiu, Y. Tao, and N. Mamoulis. The B^{dual}-tree: Indexing moving objects by space-filling curves in the dual space. *Very Large Data Bases Journal*, 2007. to appear.

60. F. Yu and V. Leung. Mobility-based predictive call admission control and bandwidth reservation in wireless cellular networks. *Computer Networks*, 38(5):577–589, 2002.

61. J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. L. Lee. Location-based spatial queries. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 443–454, 2003.

62. Z. Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: Overview and challenges. *IEEE Communications Surveys & Tutorials*, 8(1):24–37, 2006.

63. B. Zheng and D. L. Lee. Semantic caching in location-dependent query processing. In *Proceedings of the International Symposium on Advances in Spatial and Temporal Databases (SSTD)*, volume 2121 of *Lecture Notes in Computer Science*, pages 97–113, 2001.

64. J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.

Index